

[illegible]

```

CCCCCCCCC      000000    LL          LL          EEEEEEEEEE   VV          VV          TTTTTTTTTT
CCCCCCCCC      000000    LL          LL          EEEEEEEEEE   VV          VV          TTTTTTTTTT
CC           00       00    LL          LL          EE          VV          VV          TT
CC           00       00    LL          LL          EE          VV          VV          TT
CC           00       00    LL          LL          EE          VV          VV          TT
CC           00       00    LL          LL          EE          VV          VV          TT
CC           00       00    LL          LL          EE          VV          VV          TT
CC           00       00    LL          LL          EE          VV          VV          TT
CC           00       00    LL          LL          EE          VV          VV          TT
CC           00       00    LL          LL          EE          VV          VV          TT
CC           00       00    LL          LL          EE          VV          VV          TT
CCCCCCCCC      000000    LLLLLLLLLL   LLLLLLLLLL   EEEEEEEEEE   VV          VV          TT
CCCCCCCCC      000000    LLLLLLLLLL   LLLLLLLLLL   EEEEEEEEEE   VV          VV          TT
                                     ...
                                     ...
                                     ...
                                     ...

LL          111111     SSSSSSSS
LL          111111     SSSSSSSS
LL          11        SS
LL          11        SS
LL          11        SS
LL          11        SS
LL          11        SSSSSS
LL          11        SSSSSS
LL          11        SS
LL          11        SS
LL          11        SS
LL          11        SS
LLLLLLLLLLL 111111     SSSSSSSS
LLLLLLLLLLL 111111     SSSSSSSS

```

CO
VO[illegible]

```

1  COLLECTION_EVENT: Procedure      Options(Ident('V04-000'));
2
3  /*
4  /******
5  /*
6  /*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
7  /*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
8  /*  ALL RIGHTS RESERVED.
9  /*
10 /*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
11 /*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
12 /*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
13 /*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
14 /*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
15 /*  TRANSFERRED.
16 /*
17 /*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
18 /*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
19 /*  CORPORATION.
20 /*
21 /*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
22 /*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
23 /*
24 /******
25 /*
26 /*/
27
28 /*
29 /*++
30 /* FACILITY:  MONITOR Utility
31 /*
32 /* ABSTRACT:  COLLECTION_EVENT AST Routine.
33 /*
34 /*           Queued from the EXECUTE_REQUEST routine each time a
35 /*           data collection is required.
36 /*
37 /*
38 /* ENVIRONMENT:
39 /*
40 /*       VAX/VMS operating system, unprivileged user mode,
41 /*       except for certain collection routines which
42 /*       run in EXEC or KERNEL mode to access system
43 /*       data bases.
44 /*
45 /* AUTHOR:  Thomas L. Cafarella, April, 1981
46 /*
47

```



```
48 1 /*
49 1 /* MODIFIED BY:
50 1 /*
51 1 /* V03-015 TLC1085 Thomas L. Cafarella 22-Jul-1984 14:00
52 1 /* Calculate scale values for Free and Modified List bar graphs.
53 1 /*
54 1 /* V03-014 TLC1082 Thomas L. Cafarella 23-Jul-1984 11:00
55 1 /* Force error message when playing back a file containing
56 1 /* only one collection.
57 1 /*
58 1 /* V03-013 TLC1072 Thomas L. Cafarella 17-Apr-1984 11:00
59 1 /* Add volume name to DISK display.
60 1 /*
61 1 /* V03-012 TLC1066 Thomas L. Cafarella 01-Apr-1984 11:00
62 1 /* Add SYSTEM class.
63 1 /*
64 1 /* V03-011 TLC1061 Thomas L. Cafarella 18-Mar-1984 11:00
65 1 /* Identify dual-path disks by allocation class.
66 1 /*
67 1 /* V03-011 TLC1058 Thomas L. Cafarella 23-Mar-1984 10:00
68 1 /* Fix MODES class when 782 and non-782 input
69 1 /* files mixed in multi-file summary.
70 1 /*
71 1 /* V03-010 TLC1051 Thomas L. Cafarella 11-Jan-1984 11:00
72 1 /* Add consecutive number to class header record.
73 1 /*
74 1 /* V03-010 PRS1002 Paul R. Senn 29-Dec-1983 16:00
75 1 /* GLOBALDEF VALUE symbols must now be longwords;
76 1 /* Use %REPLACE rather than GLOBALDEF VALUE for any equated
77 1 /* symbols which are not 4 bytes in length;
78 1 /*
79 1 /* V03-009 TLC1050 Thomas L. Cafarella 06-Dec-1983 11:00
80 1 /* Change directory information in DLOCK class.
81 1 /*
82 1 /* V03-008 TLC1046 Thomas L. Cafarella 26-Aug-1983 18:00
83 1 /* Force flush to occur after all classes written to file.
84 1 /*
85 1 /* V03-007 TLC1040 Thomas L. Cafarella 15-Jun-1983 10:00
86 1 /* Add directory node indicator to DLOCK class.
87 1 /*
88 1 /* V03-006 TLC1035 Thomas L. Cafarella 06-Jun-1983 15:00
89 1 /* Add homogeneous class type and DISK class.
90 1 /*
91 1 /* V03-005 TLC1029 Thomas L. Cafarella 21-Apr-1983 10:00
92 1 /* Correctly calculate "Interrupt Stack" string.
93 1 /*
94 1 /* V03-004 TLC1028 Thomas L. Cafarella 14-Apr-1983 16:00
95 1 /* Add interactive user interface.
96 1 /*
97 1 /* V03-001 TLC0014 Thomas L. Cafarella 01-Apr-1982 13:00
98 1 /* Correct attached processor time reporting for MODES.
99 1 /*
100 1 /* V03-003 TLC1011 Thomas L. Cafarella 29-Mar-1982 20:00
101 1 /* Move system service names for SSERROR msg to static storage.
102 1 /*
103 1 /* V03-002 TLC1003 Thomas L. Cafarella 23-Mar-1982 13:00
```

COLLECTION_EVENT
V04-000

E 15
16-SEP-1984 02:16:56
5-SEP-1984 15:08:38

VAX-11 PL/I X2.1-273
DISK\$VMSMASTER:[MONITOR.SRC]COLLEVT.PLI;1 (2) Page 3

104 : 1 /*
105 : 1 /*
106 : 1 /*
107 : 1 /*
108 : 1 /*
109 : 1 /*
110 : 1 /*
111 : 1 /*--
112 : 1 /*/
113 : 1

Fix up module headers.

V03-001 TLC1002 Thomas L. Cafarella 20-Mar-1982 13:00
Move collection event flag to REQUEST.PLI for consolidation.

Compress bar graph range for MODES and TOPCPU.

```
114 | 1 | /*  
115 | 1 | /*  
116 | 1 | /*  
117 | 1 | /*  
118 | 1 | /*  
119 | 1 | /*  
120 | 1 | /*/  
121 | 1 |  
122 | 1 | %INCLUDE MONDEF; /* Monitor utility structure definitions */  
890 | 1 |  
891 | 1 | /*  
892 | 1 | /*  
893 | 1 | /*  
894 | 1 | /*  
895 | 1 | /*  
896 | 1 | /*  
897 | 1 | /*/  
898 | 1 |  
899 | 1 | %INCLUDE SYS$SETIMR; /* $SETIMR system service */  
908 | 1 |
```



```
909 1 /*
910 1 /*
911 1 /*
912 1 /*
913 1 /*
914 1 /*
915 1 /*
916 1
917 1 Declare
918 1 MNR$_CLASMISS FIXED BINARY(31) GLOBALREF VALUE, /* Error message */
919 1 MNR$_SSERROR FIXED BINARY(31) GLOBALREF VALUE, /* Error message */
920 1 MNR$_BEGLEND FIXED BINARY(31) GLOBALREF VALUE, /* Error message */
921 1 MNR$_COLLERR FIXED BINARY(31) GLOBALREF VALUE; /* Error message */
922 1
923 1 Declare
924 1 COLL_EV_FLAG FIXED BINARY(31) GLOBALREF VALUE, /* Collection event flag */
925 1 MAX_CLASS_NO FIXED BINARY(31) GLOBALREF VALUE, /* Maximum defined class number */
926 1 SKIP_TO_CLASS FIXED BINARY(31) GLOBALREF VALUE; /* Skip to class record indicator for READ_INPUT rtn */
927 1
928 1 Declare
929 1 COLLENDED BIT(1) ALIGNED GLOBALREF, /* YES => collection has ended */
930 1 COLL_STATUS FIXED BINARY(31) GLOBALREF, /* COLLECTION_EVENT return status code */
931 1 NORMAL FIXED BINARY(31) GLOBALREF, /* MONITOR normal return status */
932 1 MULT_TEMP FIXED BINARY(31) GLOBALREF, /* Temp hold area for MCASL_INT_MULT */
933 1 INTERVAL_DEL BIT(64) ALIGNED GLOBALREF, /* Delta time value for Interval */
934 1 SETIMR_STR FIXED BINARY(7) GLOBALREF; /* Count byte for $SETIMR cstring */
935 1
936 1 Declare
937 1 FLUSH_IND BIT(1) ALIGNED GLOBALREF, /* Flush indicator; YES=> perform FLUSH */
938 1 FLUSH_COLL$ FIXED BINARY(15) GLOBALREF, /* Number of collection events between FLUSH's */
939 1 FLUSH_CTR FIXED BINARY(15) GLOBALREF; /* Down counter for FLUSH_COLL$ */
940 1
941 1 Declare
942 1 CDBPTR POINTER GLOBALREF, /* Pointer to CDB (Class Descriptor Block) */
943 1 C POINTER DEFINED(CDBPTR), /* Synonym for CDBPTR */
944 1 MRBPTR POINTER GLOBALREF, /* Pointer to MRB (Monitor Request Block) */
945 1 M POINTER DEFINED(MRBPTR), /* Synonym for MRBPTR */
946 1 M$CAPTR POINTER GLOBALREF, /* Pointer to MCA (Monitor Communication Area) */
947 1 MC POINTER DEFINED(M$CAPTR), /* Synonym for M$CAPTR */
948 1 SPTR POINTER GLOBALREF, /* Pointer to SYI (System Information Area) */
949 1 CCDPTR POINTER GLOBALREF; /* Pointer to CCD (Current Class Descriptor) Array */
950 1
951 1 Declare
952 1 INPUT_FILE FILE RECORD INPUT, /* Monitor Input (Playback) File */
953 1 INPUT_CPTR POINTER GLOBALREF, /* Ptr to input buffer count word */
954 1 INPUT_DATA CHAR(512) VARYING BASED(INPUT_CPTR); /* Playback file input buffer */
955 1
956 1 Declare
957 1 01 CURR_CLASS_DESCR (MAX_CLASS_NO+1) BASED(CCDPTR), /* Current Class Descriptor */
958 1 /* This array of structures includes a CCD (Current
959 1 /* Class Descriptor) for each possible class */
960 1 02 CURR_CDBPTR POINTER, /* CDBPTR for current class */
961 1 02 CURR_CLASS_NO FIXED BINARY(7); /* Class number for current class */
962 1
```

```
963 1 /*
964 1 /*
965 1 /*
966 1 /* GLOBAL STORAGE DEFINITIONS
967 1 /*
968 1 /*
969 1 /*/
970 1
971 1 /*
972 1 /*
973 1 /*
974 1 /* COMPILE-TIME CONSTANTS
975 1 /*
976 1 /*
977 1 /*/
978 1
979 1 %REPLACE NOT_SUCCESSFUL BY '0'B; /* Failing status bit */
980 1 %REPLACE YES BY '1'B; /* For general use */
981 1 %REPLACE NO BY '0'B; /* For general use */
982 1 /*
983 1 /*
984 1 /*
985 1 /* OWN STORAGE
986 1 /*
987 1 /*
988 1 /*/
989 1
990 1 Declare
991 1 CALL FIXED BINARY(31), /* Holds function value (return status) of called ro
992 1 STATUS BIT(1) BASED(ADDR(CALL)), /* Low-order status bit for called routines */
993 1 I FIXED BINARY(15), /* Index for DO loop */
994 1 BUFF_PTR POINTER, /* Temporary pointer to input file buffer */
995 1 CURR_TYPE FIXED BINARY(7), /* Class record type of record just read */
996 1 PREV_TYPE FIXED BINARY(7), /* Class record type of record previously read */
997 1 PREV_CONT BIT(1) ALIGNED, /* Value of MNR_CLASSV_CONT for record previously rea
998 1 CLASS_MISSING BIT(1) ALIGNED, /* For Playback, ON => requested class not in file *
999 1 CLASS_FOUND BIT(1) ALIGNED; /* For Playback, ON => requested class found in file
1000 1
1001 1 Declare
1002 1 MON_ERR ENTRY (ANY VALUE, ANY, ANY) OPTIONS(VARIABLE), /* MONITOR MACRO-32 routine to log synchronous error
1003 1 READ_INPUT ENTRY (FIXED BINARY(7)), /* MONITOR routine to read an input (playback) file
1004 1 COLLECTION_END ENTRY, /* MONITOR routine to indicate end of collection */
1005 1 CLASS_COLLECT ENTRY (FIXED BINARY(7)) /* MONITOR MACRO-32 routine to collect a buffer of d
1006 1 RETURNS(FIXED BINARY(31));
1007 1
```



```
1008 1 /*
1009 1 /*++
1010 1 /*
1011 1 /* FUNCTIONAL DESCRIPTION:
1012 1 /*
1013 1 /*     COLLECTION_EVENT
1014 1 /*
1015 1 /*     COLLECTION_EVENT is an AST routine invoked via the $DCLAST
1016 1 /*     system service from the EXECUTE_REQUEST routine, or via
1017 1 /*     the $SETIMR system service from a previous invocation of
1018 1 /*     COLLECTION_EVENT. It performs performance data collection
1019 1 /*     from VMS data bases of the running system, or from an
1020 1 /*     input recording file. A single invocation of COLLECTION_EVENT
1021 1 /*     causes collection of data for all classes in the MONITOR
1022 1 /*     request. The data is collected by calling the CLASS_COLLECT
1023 1 /*     routine once for each class. CLASS_COLLECT stores the data in a
1024 1 /*     collection buffer (pointed to by the CDB) for each class.
1025 1 /*
1026 1 /*     On the first collection event, class-specific initialization
1027 1 /*     is performed by a call to the CLASS_INIT routine.
1028 1 /*
1029 1 /* INPUTS:
1030 1 /*
1031 1 /*     None
1032 1 /*
1033 1 /* IMPLICIT INPUTS:
1034 1 /*
1035 1 /*     ALL MONITOR variables accessible to this routine.
1036 1 /*
1037 1 /* OUTPUTS:
1038 1 /*
1039 1 /*     None
1040 1 /*
1041 1 /* IMPLICIT OUTPUTS:
1042 1 /*
1043 1 /*     MCASL_COLLCNT is incremented by 1.
1044 1 /*
1045 1 /*     Data for all requested classes has been collected into
1046 1 /*     their respective CDB collection buffers.
1047 1 /*
1048 1 /*     ALL MONITOR variables accessible to this routine.
1049 1 /*
1050 1 /* ROUTINE VALUE:
1051 1 /*
1052 1 /*     COLL_STATUS contains the status of this collection event upon
1053 1 /*     exit.
1054 1 /*
1055 1 /* SIDE EFFECTS:
1056 1 /*
1057 1 /*     If this is the final collection event, the COLLENDED bit is set.
1058 1 /*
1059 1 /*--
1060 1 /*/
1061 1
```

```
1062 1 IF COLLEDED = YES THEN RETURN; /* If collection has already ended, return immediate
1063 1
1064 1 IF M->MRBSV_PLAYBACK /* Playback Request */
1065 1 THEN DO;
1066 2 IF MC->MCASL_COLLCNT = 0 /* If first collection event, */
1067 2 THEN MULT_TEMP = 1; /* ... set multiple to trigger on this collection */
1068 2 MC->MCASV_MULTFND = NO; /* Indicate multiple not yet found */
1069 2 MULT_TEMP = MULT_TEMP - 1; /* Count down toward zero */
1070 2 IF MULT_TEMP = 0 /* If it's time to record and display, */
1071 2 THEN DO;
1072 3 MC->MCASV_MULTFND = YES; /* ... indicate so */
1073 3 MULT_TEMP = MC->MCASL_INT_MULT; /* ... and re-load multiple value for next collectio
1074 3 MC->MCASL_CONSEC_REC = MC->MCASL_CONSEC_REC + 1; /* ... also update to a new consec no for recordi
1075 3 END;
1076 3 BUFF_PTR = MC->MCASA_INPUT_PTR; /* Get pointer to input file buffer for later use */
1077 3 PREV_TYPE = -1; /* Dummy previous record type (class no) */
1078 3 PREV_CONT = NO; /* Dummy previous "continue" bit setting */
1079 3 CLASS_MISSING = '0'B; /* Class not missing */
1080
1081 3 DO I = 1 TO M->MRBSW_CLASSCT WHILE (^ MC->MCASV_EOF & ^ CLASS_MISSING); /* Loop through all requeste
1082 3 CLASS_FOUND = '0'B; /* Haven't found class yet */
1083 3 CDBPTR = CURR_CDBPTR(1); /* Set up current CDB */
1084 3 IF MC->MCASL_COLLCNT = 0 /* If first collection event */
1085 3 THEN CALC = CLASS_INIT(); /* ... then do init for this class */
1086
1087 3 DO WHILE (^ MC->MCASV_EOF & ^ CLASS_FOUND & ^ CLASS_MISSING); /* Loop causes input file to skip past unwan
1088 3 /* ... classes within the recorded interval
1089 3 CURR_TYPE = BUFF_PTR->MNR_CLSSB_TYPE; /* Get class (record) type of current record */
1090 3 IF (CURR_TYPE < PREV_TYPE) | (CURR_TYPE > CURR_CLASS_NO(I)) | /* Check for missing class (should never occur) */
1091 3 (CURR_TYPE = PREV_TYPE & PREV_CONT = NO)
1092 3 THEN DO;
1093 4 CLASS_MISSING = YES; /* Indicate "class missing" error */
1094 4 COLL_STATUS = MNR$_CLSMISS; /* Save failing status */
1095 4 CALL MON_ERR(MNR$_CLSMISS); /* ... and log the error */
1096 4 END;
1097 4 ELSE DO;
1098 5 IF CURR_TYPE = CURR_CLASS_NO(I) /* If inputted class = monitored class */
1099 5 THEN DO;
1100 6 CLASS_FOUND = YES; /* Indicate found the record needed */
1101 6 CALL = CLASS_COLLECT(CURR_CLASS_NO(I)); /* Collect data for this class */
1102 6 IF STATUS = NOT_SUCCESSFUL /* If collection failed, */
1103 6 THEN DO;
1104 7 COLL_STATUS = MNR$_COLLERR; /* Save failing status */
1105 7 CALL MON_ERR(MNR$_COLLERR,CALL); /* Log the error */
1106 7 CALL COLLECTION_END(); /* ... and terminate collection */
1107 7 END;
1108 6 END;
1109 5 PREV_TYPE = CURR_TYPE; /* Current now becomes previous */
1110 5 PREV_CONT = BUFF_PTR->MNR_CLSSV_CONT; /* Save previous "continue" bit setting */
1111 5 CALL READ_INPUT(SKIP_TO_CLASS); /* Read the next class record */
1112 5 END;
1113 4 END;
1114
1115 3 IF MC->MCASV_EOF | CLASS_MISSING /* If anything but CLASS FOUND, */
1116 3 THEN CALC COLLECTION_END(); /* ... then indicate collection ended */
1117
```

COLLECTION_EVEN
V04-000

1118 3
1119 2

END;

K 15
16-SEP-1984 02:16:58
5-SEP-1984 15:08:38

VAX-11 PL/I X2.1-273
DISK\$VMSMASTER:[MONITOR.SRC]COLLEVT.PLI;1 (7)

Page 9

GE
VO


```

1120      IF COLLENDED = NO                                /* If end of collection not indicated, then scan */
1121      THEN DO;                                           /* ... the input file for the beginning of the next
1122      CURR_TYPE = BUFF_PTR->MNR_CLSSB_TYPE;              /* ... interval and leave the file positioned there.
1123      DO WHILE(^ MC->MCASV_EOF & CURR_TYPE > PREV_TYPE); /* Get class (record) type of current record */
1124      PREV_TYPE = CURR_TYPE;                             /* Loop while class type numbers increase */
1125      CALL_READ_INPUT(SKIP TO CLASS);                    /* Current now becomes previous */
1126      CURR_TYPE = BUFF_PTR->MNR_CLSSB_TYPE;              /* Read the next class record */
1127      END;                                                /* Get class (record) type of current record */
1128      IF MC->MCASV_EOF                                    /* If end-of-file reached, */
1129      THEN CALL COLLECTION_END();                        /* ... then indicate so */
1130      END;
1131
1132      IF MC->MCASV_EOF & MC->MCASL_COLLCNT = 0           /* If end-of-file after first collection event, */
1133      THEN DO;                                           /* ... then this is an error */
1134      COLL_STATUS = MNRS_BEGNLEND;                      /* Save failing status */
1135      CALL_MON_ERR(MNRS_BEGNLEND);                      /* ... and log the error */
1136      END;
1137
1138      END;                                               /* End of Playback Request processing */
1139
1140
1141
1142

```

```
1143 1 ELSE DO; /* Live Request */
1144 2
1145 2 MC->MCASL_CONSEC_REC = MC->MCASL_CONSEC_REC + 1; /* Update to a new consec no for recording */
1146 2 IF M->MRBSV_RECORD /* If recording, */
1147 2 THEN FLUSH_CTR = FLUSH_CTR - 1; /* Decrement flush counter for this coll event */
1148 2
1149 2 DO I = 1 TO M->MRBSW_CLASSCT WHILE (COLLENDED = NO); /* Loop once for each requested class */
1150 2 CDEPTR = CURR_CDBPTR(I); /* Set up current CDB */
1151 2 IF MC->MCASL_COLLCNT = 0 /* If first collection event */
1152 2 THEN CALL = CLASS_INIT(); /* ... then do init for this class */
1153 2
1154 2 IF FLUSH_CTR = 0 & CURR_CLASS_NO(I) = MC->MCASB_LASTC /* If FLUSH_CTR reached zero, and this is last cl
1155 2 THEN DO; /* ... then time to flush */
1156 2 FLUSH_IND = YES; /* Indicate flush required */
1157 2 FLUSH_CTR = FLUSH_COLLIS; /* ... and start down counter at beginning again */
1158 2 END;
1159 2
1160 2 CALL = CLASS_COLLECT(CURR_CLASS_NO(I)); /* Collect data for this class */
1161 2 IF STATUS = NOT_SUCCESSFUL /* If collection failed, */
1162 2 THEN DO;
1163 2 COLL_STATUS = MNR$_COLLERR; /* Save failing status */
1164 2 CALL_MON_ERR(MNR$_COLLERR,CALL); /* Log the error */
1165 2 CALL COLLECTION_END(); /* ... and terminate collection */
1166 2 END;
1167 2
1168 2 END;
1169 2
1170 2 IF COLLENDED = NO /* If not at end of collection, */
1171 2 THEN DO;
1172 2 CALL = SYS$SETIMR(COLL_EV_FLAG,INTERVAL_DEL,COLLECTION_EVENT); /* Re-enter COLLECTION_EVENT at specified interval */
1173 2 /* COLL_EV_FLAG is not used; it is just a dummy */
1174 2 IF STATUS = NOT_SUCCESSFUL /* If $SETIMR failed, */
1175 2 THEN DO;
1176 2 COLL_STATUS = MNR$_SSERROR; /* Save failing status */
1177 2 CALL_MON_ERR(MNR$_SSERROR,CALL,SETIMR_STR); /* Log the error */
1178 2 CALL COLLECTION_END(); /* ... and terminate collection */
1179 2 END;
1180 2
1181 2 END; /* End of Live Request processing */
1182 1
1183 1 MC->MCASL_COLLCNT = MC->MCASL_COLLCNT + 1; /* Count this collection event */
1184 1 RETURN; /* Return to caller */
1185 1
```

```

1186 1 CLASS_INIT: Procedure Returns(fixed binary(31)); /* Class-specific initialization */
1187
1188 /*
1189 /**+
1190 /*
1191 /* FUNCTIONAL DESCRIPTION:
1192 /*
1193 /* CLASS_INIT
1194 /*
1195 /* This routine is called by COLLECTION_EVENT on the first
1196 /* collection event to perform class-specific initialization.
1197 /* Currently, the MODES, PROCESSES, DISK, DLOCK and SYSTEM
1198 /* classes require such initialization.
1199 /*
1200 /* INPUTS:
1201 /*
1202 /* None
1203 /*
1204 /* OUTPUTS:
1205 /*
1206 /* None
1207 /*
1208 /* IMPLICIT OUTPUTS:
1209 /*
1210 /* Initialization for the MODES, PROCESSES, DISK, DLOCK and SYSTEM classes performed.
1211 /*
1212 /* ROUTINE VALUE:
1213 /*
1214 /* $$$_NORMAL
1215 /*
1216 /* SIDE EFFECTS:
1217 /*
1218 /* None
1219 /*
1220 /*--
1221 /*/
1222

```



```
1223  /*
1224  /*
1225  /*
1226  /*
1227  /*
1228  /*
1229  /*
1230  /*
1231  Declare
1232  PROCESSES CLSNO    FIXED BINARY(31) GLOBALREF VALUE,          /* PROCESSES class number */
1233  MODES CLSNO        FIXED BINARY(31) GLOBALREF VALUE,          /* MODES class number */
1234  DISK CLSNO         FIXED BINARY(31) GLOBALREF VALUE,          /* DISK class number */
1235  DLOCK CLSNO        FIXED BINARY(31) GLOBALREF VALUE,          /* DLOCK class number */
1236  SYSTEM CLSNO       FIXED BINARY(31) GLOBALREF VALUE,          /* SYSTEM class number */
1237  TOP_RANGE          FIXED BINARY(31) GLOBALREF VALUE,          /* Range value for TOPB, TOPD, TOPF bar graph */
1238  MODES_STRLEN       FIXED BINARY(31) GLOBALREF VALUE;          /* Length of "Interrupt Stack" string */
1239
1240  Declare
1241  IDBPTR             POINTER,                                     /* Pointer to Item Descriptor Block (IDB) */
1242  ITMSTR (1:C->CDB$L_ICOUNT) BIT(8) ALIGNED BASED(C->CDB$A_ITMSTR), /* Vector of item numbers for this class */
1243  ITEM_IDX           FIXED BINARY(15),                          /* Index into IDB_BLOCK */
1244  ITEMNO             FIXED BINARY (7);                          /* Item number used in DO loop */
1245
1246  Declare
1247  1 PERFTABLE GLOBALREF,                                       /* Table of IDB's */
1248  2 IDB_BLOCK (0:255) CHAR(IDB$K_ILENGTH);                   /* Up to 256 IDB's */
1249
1250  Declare
1251  1 PINTERRUPT STR BASED(IDBPTR->IDB$A_LNAME),                /* Counted string for "Interrupt Stack PRIMARY" */
1252  2 L FIXED BINARY(7),                                         /* Count */
1253  2 S CHAR(1);                                                 /* First character of string */
1254
1255  Declare
1256  REVLEVELS          (0:127) FIXED BINARY(7) GLOBALREF;      /* Revision Levels Vector */
1257
1258  Declare
1259  1 DIR_STR BASED(IDBPTR->IDB$A_LNAME),                        /* Counted string for "Directory" items in DLOCK */
1260  2 L FIXED BINARY(7),                                         /* Count */
1261  2 X CHAR(17),                                                /* Uninteresting characters of string */
1262  2 S CHAR(5);                                                 /* Characters of interest */
1263
1264  Declare
1265  PROCTITLE (0:127) GLOBALREF POINTER;                        /* Table of pointers to PROCESSES screen titles */
1266
1267  Declare
1268  1 BU_SYS_SINGLE     GLOBALREF,                                /* Bar graph range values for SYSTEM class (single s
1269  2 BSS_RANGE (1:17)  FIXED BINARY(31);
1270
```

```
1271 IF CURR_CLASS_NO(I) = MODES_CLSNO
1272 THEN DO;
1273   C->CDB$V_CPU_COMB = NO;
1274   MC->MC$A_MPADDR = NULL();
1275   UNSPEC(ITEM_IDX) = ITMSTR(1);
1276   IDBPTR = ADDR(IDB_BLOCK(ITEM_IDX));
1277   PINTERRUPT_STR.L = MODES_STLEN;
1278   IF SPTR->MNR_SYISB_MPCPU5 = 2
1279   THEN DO;
1280     C->CDB$L_ICOUNT = C->CDB$L_ICOUNT *
1281     SPTR->MNR_SYISB_MPCPU5;
1282     C->CDB$W_BLKLEN = C->CDB$W_BLKLEN *
1283     SPTR->MNR_SYISB_MPCPU5;
1284     IF C->CDB$V_CPU & M->MRBSV_SYSCLS = NO & M->MRBSV_MFSUM = NO /* If CPU-specific display requested
1285     THEN DO; /* AND SYSTEM class not being monitored, */
1286       C->CDB$L_ECOUNT = C->CDB$L_ICOUNT; /* AND not multi-file summary, */
1287       END; /* Increase number of displayed elements */
1288     ELSE DO; /* Combined display */
1289       C->CDB$V_CPU_COMB = YES; /* Indicate that collected items must be */
1290       PINTERRUPT_STR.L = PINTERRUPT_STR.L - 10; /* ... combined for display */
1291       END; /* Shorten "Interrupt Stack" display string
1292     /* ... to remove the word "PRIMARY" */
1293   ELSE
1294     PINTERRUPT_STR.L = PINTERRUPT_STR.L - 10; /* Uniprocessor system */
1295   /* Shorten "Interrupt Stack" display string
1296   /* ... to remove the word "PRIMARY" */
1297   END;
1298 IF CURR_CLASS_NO(I) = PROCS_CLSNO
1299 THEN DO;
1300   C->CDB$A_TITLE = PROCTITLE(C->CDB$B_ST);
1301   IF C->CDB$B_ST = TOPC_PROC
1302   THEN C->CDB$L_RANGE = 100;
1303   ELSE C->CDB$L_RANGE = TOP_RANGE;
1304   END;
1305
1306
1307
1308
1309
1310
1311
```

```
1312      IF CURR_CLASS NO(1) = DISK_CLSNO
1313      & REVLEVELS(DISK_CLSNO) >= 0
1314      THEN DO;
1315          C->CDB$V_DISKAC = YES;
1316          IF REVLEVELS(DISK_CLSNO) > 1
1317          THEN C->CDB$V_DISKVN = YES;
1318          ELSE C->CDB$V_DISKVN = NO;
1319      END;
1320      ELSE DO;
1321          C->CDB$V_DISKAC = NO;
1322          C->CDB$V_DISKVN = NO;
1323      END;
1324
1325      IF CURR_CLASS NO(1) = DISK_CLSNO
1326      & M->MRB$V_MFSUM = NO
1327      & C->CDB$B_ST = ALL_STAT
1328      THEN C->CDB$V_WIDE = YES;
1329      ELSE C->CDB$V_WIDE = NO;
1330
1331      IF CURR_CLASS NO(1) = DLOCK_CLSNO
1332      & REVLEVELS(DLOCK_CLSNO) = 0
1333      THEN DO;
1334          DO ITEMNO = 13 TO 15 BY 1;
1335              UNSPEC(ITEM_IDX) = ITMSTR(ITEMNO);
1336              IDBPTR = ADDR(IDB_BLOCK(ITEM_IDX));
1337              IF SPTR->MNR_SYISV_RESERVED1
1338              THEN DIR_STR.S = 'Incom';
1339              ELSE DIR_STR.S = 'Outgo';
1340          END;
1341      END;
1342
1343      END;
1344
1345      END;
1346
1347      END;
1348
```

```
/* If DISK class ... */
/* ... AND it is any rev level after 0, */
/* then indicate DISK with allocation clas
/* If any rev level after 1, */
/* then indicate DISK with volume names
/* else indicate not */
/* else indicate no alloc class in name, *
/* ... and no volume name */
/* If DISK class ... */
/* ... AND it's not m.f. summary, */
/* ... AND all stats requested, */
/* then indicate wide display, */
/* else indicate usual width */
/* If DLOCK class ... */
/* ... AND it is rev level 0, */
/* Change text for last three items */
/* Zero-extend ITMSTR element to word */
/* Set up IDB ptr in order to */
/* ... reference DIR_STR */
/* If this is directory node, */
/* then rates are 'Incoming' */
/* else they are 'Outgoing' */
/* End of DO loop */
```


COLLECTION_EVENT
V04-000

E 16
16-SEP-1984 02:17:01
5-SEP-1984 15:08:38

VAX-11 PL/I X2.1-273 Page 16
ISK\$VMSMASTER:[MONITOR.SRC]COLLEVT.PL1;1 (14)

```
1349      2      IF CURR_CLASS_NO(I) = SYSTEM_CLSNO      /* If SYSTEM class, */
1350      2      THEN DO:
1351      2          BSS_RANGE(14) = SPTR->MNR_SYISL_BALSETMEM; /* Stash away range of Free List bar graph */
1352      2          BSS_RANGE(15) = SPTR->MNR_SYISL_MPWHILIM; /* ... and Modified List bar graph */
1353      2          END;
1354      2
1355      2      RETURN(NORMAL);
1356      2
1357      2      END CLASS_INIT;
1358      2
1359      1      END COLLECTION_EVENT;
```

COMMAND LINE

PLI/LIS=LISS:COLLEVT/OBJ=OBJ\$:COLLEVT MSRC\$:COLLEVT+LIB\$:MONLIB/LIB

0239

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY